Implementasi Algoritma Pencarian Shannon Type-A Pada Program Permainan Catur

Dian Rachmanto, ST, Waru Djuriatno, S.T., M.T., dan Ir. Muhammad Aswin, M.T.
Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya
cressyfrost@hotmail.com

Abstrak-- Pencarian adalah metode untuk melihat kedepan pada setiap gerakan dan mengevaluasi posisi setelah melakukan gerakan tersebut. Shannon Type A adalah sebuah brute-force search yang melihat seluruh kemungkinan dengan kedalaman yang sudah ditentukan.

Strategi Type-A dikemukakan oleh Claude Shannon pada publikasinya yang sangat terkenal: *Programming a Computer for Playing Chess* [1]. sebagai strategi brute-force. Algoritma Type-A cukuplah sederhana, namun membutuhkan komputasi yang teramat besar (tergantung jumlah kedalaman pencarian).

Algoritma ini bekerja dengan menggunakan prinsip minimax, yaitu sebuah prinsip yang berusaha untuk memperoleh keuntungan (gain) sebesar-besarnya dengan kerugian (loss) yang sekecil-kecilnya. Proses pencarian bekerja dengan menerapkan prinsip tersebut hingga jumlah kedalaman yang telah ditentukan sebelumnya.

Kata kunci: Pemrograman Catur, Algoritma Pencarian, Shannon Type-A, Minimax, Permainan Catur

I. PENDAHULUAN

Pemrograman Catur dalam bentuk digital telah ada sejak tahun 1950, sesaat setelah komputer elektronik pertama dibuat. Sejak itu pemrograman catur telah berkembang secara signifikan, hingga pada tahap mesin dan program bisa bermain seperti Grand Master Chess.

Pemikiran untuk membuat mesin yang bisa bermain catur telah ada sejak abad 18. Tetapi yang pertama membuat sebuah ide untuk membuat program komputer yang lengkap dan menanamkan pondasi teori yang digunakan hingga saat ini adalah Claude Shannon.

Shannon menganggap catur adalah bidang percobaan yang bagus, walaupun cukup berat untuk mengembangkan konsep itu pada era tersebut. Catur beroperasi dengan simbol, tidak hanya dengan angka saja. Memerlukan pengambilan keputusan, tidak hanya mengikuti baris-baris program yang telah disusun, memilih solusi tidak hanya antara "baik" atau "buruk", tetapi juga member nilai atau kualitas dengan beberapa ukuran. Catur memenuhi semua persyaratan tersebut, yaitu membutuhkan beberapa "pemikiran" untuk level permainan yang masuk akal, dan bukan pekerjaan yang mudah. Dan juga catur mempunyai beberapa properti yang cocok untuk implementasi dalam komputer: sebuah permainan catur selalu berakhir, selalu beroperasi pada peraturan yang ada, dan tidak terlalu kompleks.

Walaupun perkembangan catur pada saat itu terlihat seperti membuang waktu, computing power dan potensi manusia, Shannon menyadari bahwa solusi yang dihasilkan dari "bidang percobaan" ini bisa dikembangkan dalam banyak aplikasi untuk hal-hal yang mirip tetapi mempunyai nilai yang lebih signifikan, seperti merancang filter atau switch, alat pembuat keputusan strategi militer atau bahkan melodi music dan masih banyak lagi. Shannon dan hasil percobaannya juga telah membuat framework dasar dan banyak konsep tentang implementasi catur pada komputer. Kebanyakan masih digunakan hingga sekarang (seperti konsep fungsi evaluasi).

Ide pengembangan catur untuk pelatihan ilmu komputer lain banyak menarik minat para ilmuwan, insinyur dan para hobbyist. Sekarang sudah banyak pengembangan catur komputer yang tersedia, dan karena ada beberapa yang mempunyai level permainan maksimum hampir sama dengan Grand Master, maka hal ini digunakan sebagai lawan bermain oleh jutaan pemain catur di seluruh dunia.

II. TINJAUAN PUSTAKA

Perancangan aplikasi secara umum merupakan langkah awal sebagai acuan untuk pembuatan aplikasi lebih lanjut. Perancangan ini didahului dengan menentukan langkah – langkah paling dasar pada aplikasi permainan catur.

Aplikasi ini dibagi menjadi 3 bagian utama, yaitu *board representation*, *move generation* dan *evaluation*.

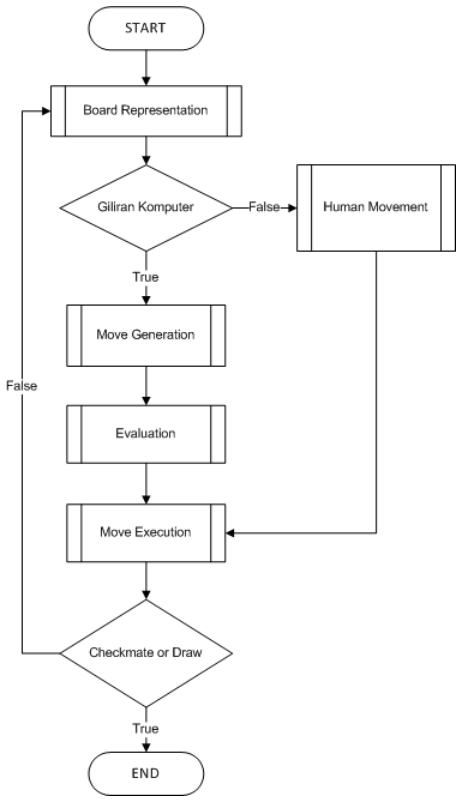
Board Representation, yaitu pembuatan seluruh variabel yang nantinya akan digunakan, beserta tampilannya didalam antarmuka (contoh: a1 untuk *White Pawn*).

Move Generation yaitu memasukkan aturan-aturan yang berlaku pada catur kedalam program, baik pseudo-legal maupun legal. Setelah itu dilakukan proses *debugging* untuk tiap bidak apakah sudah memenuhi aturan atau tidak.

Evaluation yaitu proses perhitungan untuk seluruh gerakan yang mungkin dilakukan, dalam hal ini seluruh gerakan yang boleh dilakukan dalam catur (sesuai no. 2). Dan setelah diketahui gerakan apa saja yang mungkin akan dilakukan proses penilaian (evaluasi) untuk setiap gerakan tersebut.

Move Execution yaitu Eksekusi gerakan sesungguhnya, kemudian mengakhiri giliran.

Proses ini bisa digambarkan pada diagram di bawah ini :



Gambar 1. Cara Kerja Sistem

Cara kerja aplikasi ini dimulai dengan input gerakan dari pemain manusia. Dalam program ini dibatasi pemain manusia hanya bisa bermain sebagai putih. Setelah papan selesai dinisialisasi, Maka Putih bisa langsung memulai permainan dengan menggerakkan bidak putih dengan cara *dragging* (menggeser) bidak putih ke petak tujuan.

Selanjutnya gerakan tersebut akan divalidasi oleh fungsi didalam program, apakah sudah sesuai peraturan atau belum (contoh: tidak ada bidak yang bisa melompati bidak lain kecuali kuda). Setelah itu dilakukan proses pengecekan apakah ada bidak teman (*allied pieces*) di petak tujuan tadi.

Jika seluruh pengecekan diatas sudah dilakukan dan semua syarat terpenuhi, maka dilakukan proses perpindahan bidak, yaitu dengan memasukkan nilai *null* pada petak asal dan merubah nilai petak tujuan dengan nilai bidak yang digerakkan. Setelah proses ini selesai, dilakukan perubahan nilai pada variabel *turn* yang menyatakan giliran, sehingga giliran berpindah pada Hitam (computer).

Pada saat giliran berpindah pada hitam akan dijalankan fungsi untuk menemukan seluruh gerakan yang mungkin untuk hitam pada keadaan papan saat ini (karena kondisi papan berubah setelah Putih melakukan gerakan tadi). Lalu seluruh gerakan tadi disimpan kedalam sbeuah variabel (contoh nilai nya adalah "2,7-2,6"). Dimana angka yang dimasukkan kedalam variabel tersebut merupakan koordinat pada papan.

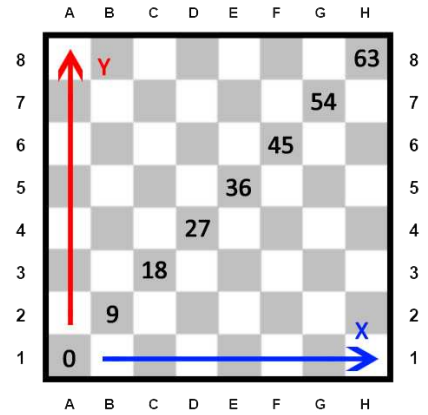
Setelah seluruh gerakan yang mungkin selesai dicatat akan dilakukan proses evaluasi untuk setiap gerakan tersebut (sisa gerakan yang jelas tidak mungkin dilakukan tidak akan dievaluasi). Proses evaluasi inilah yang menggunakan algoritma Shannon Type-A yang akan dijelaskan lebih lanjut.

Setelah semua gerakan selesai dievaluasi akan dipilih gerakan yang akan menghasilkan nilai terbaik untuk hitam (paling menguntungkan). Lalu gerakan tersebut akan dieksekusi dengan cara yang sama seperti Putih, hal ini akan terus diulangi hingga terjadi *checkmate* (skakmat) atau *draw* (seri/remis).

III. METODOLOGI

Board Representation

Board Representation yang digunakan pada program ini adalah Array 2-Dimensi berukruan 8x8. Dikarenakan jumlahnya yang sama dengan jumlah petak (squares) pada papan catur.



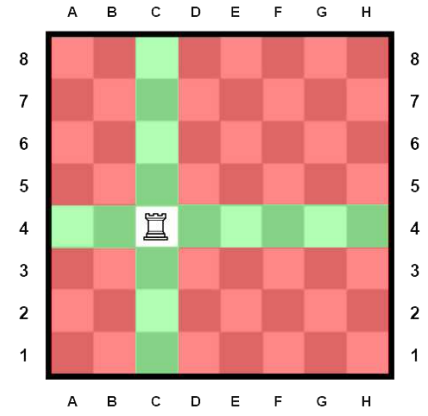
Gambar 2. *Board Representation*

Pada gambar diatas, Array dimulai dari pojok kiri bawah, seperti system koordinat Cartesian. Yaitu index pertama atau [0][0] berada pada koordinat A,1 dan index terakhir atau [7][7] berada pada koordinat H,8.

Langkah selanjutnya adalah memberi nilai untuk setiap bidak, yang nantinya akan digunakan untuk evaluasi.

Move Generation

Move Generation berlaku baik untuk Putih maupun Hitam. Proses ini bertujuan untuk memeriksa legalitas dari gerakan yang akan dilakukan. Legalitas yang dimaksud adalah bisa atau tidaknya suatu bidak akan bergerak pada petak tujuan sesuai dengan peraturan yang berlaku pada catur.



Gambar 3. *Move Generation*

Dalam program ini *Move Generation* akan dipecah menjadi satu peraturan global, dan sisanya adalah peraturan untuk setiap jenis bidak, dengan rincian:

Peraturan Global
Terdiri dari dua bagian yaitu:

- ✓ Gerakan tidak akan bisa dilakukan jika petak tujuan atau jalan menuju petak tujuan sudah ditempati oleh bidak dengan warna yang sama (*allied pieces*). Dengan pengecualian untuk Knight yang bisa "melompat".

- ✓ Gerakan tidak akan bisa dilakukan jika gerakan tersebut akan mengakibatkan bidak "King" player tersebut berada dalam posisi skak (*check*).

Kedua hal tersebut berlaku untuk semua bidak tanpa kecuali walaupun sudah memenuhi peraturan individual untuk semua jenis bidak.

Evaluation

Setiap gerakan yang mungkin harus dianalisa dan dinilai berdasarkan perubahan papan yang terjadi akibat langkah tersebut. Setelah seluruh evaluasi selesai, dipilih gerakan yang paling bagus, yaitu gerakan yang akan menghasilkan hasil evaluasi paling tinggi.

Proses Evaluasi akan dijalankan saat *turn* (giliran) berpindah ke computer. Komputer akan mengevaluasi papan berdasarkan kondisi saat ini (yaitu setelah pemain sebelumnya *selesai* melakukan gerakan).

Lalu dilakukan pengecekan apakah kondisi saat ini merupakan skakmat atau remis atau bukan. Karena jika terjadi skakmat atau remis permainan akan berakhir sampai disini, sehingga pemain selanjutnya (baik manusia ataupun computer) tidak akan mendapat giliran.

1. Pengecekan seluruh bidak dengan warna yang sama.
Pengecekan ini dilakukan dengan tujuan untuk menemukan bidak apa saja yang berwarna sama dengan warna pemain saat ini beserta lokasinya.

Hal ini dikarenakan kita hanya bisa menggerakkan bidak milik kita sendiri (yang berwarna sama). Misalnya jika dalam suatu elemen array terdapat suatu huruf 'b' maka bisa dipastikan dalam index ini terdapat bidak hitam (dalam hal ini b adalah symbol untuk Black). Jika terdeteksi suatu bidak hitam dalam Loop tersebut, akan langsung dipanggil fungsi untuk melakukan proses selanjutnya.

2. Pengecekan apakah bidak yang ditemukan bisa bergerak.
Proses ini dilakukan untuk memeriksa apakah bidak yang ditemukan bisa bergerak atau tidak, karena walaupun bidak tersebut adalah milik pemain tersebut, belum tentu bidak tersebut bisa bergerak karena bisa saja bidak tersebut tertutup (*blocked*) oleh bidak lain ataupun jika pergerakan bidak tersebut akan mengakibatkan King pemain tersebut berada dalam kondisi skak.

Setelah itu akan dilakukan proses satu kali Loop lagi, yang berfungsi untuk menentukan petak tujuan bidak tersebut akan bergerak.

Proses pengecekan dibagi menjadi dua bagian yaitu:

- ✓ Memeriksa apakah bidak tersebut bisa bergerak ke petak tujuan berdasarkan peraturan yang telah dibuat sebelumnya (Move Generation).

- ✓ Memeriksa apakah jika gerakan tersebut dilakukan akan mengakibatkan posisi King pemain tersebut berada dalam kondisi skak.

Jika sudah diperiksa gerakan tersebut boleh dilakukan menurut peraturan dan gerakan tersebut tidak membahayakan keselamatan King pemain tersebut maka akan dilakukan pengisian data kedalam.

Array tersebut berfungsi untuk mencatat data dari suatu gerakan yang boleh untuk dilakukan, dalam hal ini data yang akan dimasukkan kedalam Array tersebut adalah koordinat dari suatu bidak dan koordinat tujuan dimana bidak tersebut boleh bergerak (yang telah divalidasi melalui proses-proses sebelumnya)

Proses ini akan terus berulang hingga seluruh bidak yang berwarna sama telah diperiksa kemungkinan bergeraknya.

3. Pemberian nilai untuk gerakan yang sudah dicatat.
Setelah seluruh gerakan yang legal untuk dilakukan selesai dicatat kedalam Array dalam proses selanjutnya, akan dilakukan proses pemberian nilai untuk mencari gerakan mana yang akan paling menguntungkan, dalam hal ini akan menghasilkan nilai tertinggi dengan eksekusi kode dibawah ini:

Looping utama dilakukan berdasarkan jumlah gerakan legal yang telah ditemukan pada proses sebelumnya. Setelah itu akan dilakukan perubahan sementara pada papan sesuai data yang diberikan oleh gerakan dalam Array sebelumnya. Eksekusi sementara dari gerakan ini bertujuan untuk menerapkan algoritma Shannon Type-A atau disebut juga metode minimax.

4. Penerapan Algoritma Shannon Type-A.
Penerapan algoritma ini bisa dianalogikan dengan tiga langkah dasar Minimax berikut:

1. *Saya akan mencoba melakukan gerakan ini, dan akan melihat bagaimana kira-kira respon dari lawan saya.*

2. *Jika gerakan tadi sudah saya lakukan, saya sekarang akan berpikir seolah-olah saya adalah lawan saya.*

3. *Dalam keadaan saat ini saya akan memilih gerakan yang paling menguntungkan untuk saya.*

Dari ketiga langkah tersebut, setelah langkah c) dilakukan maka akan diketahui:

"*Jika saya jadi melakukan langkah tadi, maka saya akan tahu hal terburuk yang akan terjadi sebagai akibat dari langkah saya tersebut*".

Sehingga keluaran dari fungsi ini adalah prediksi dari langkah yang akan diambil oleh lawan.

Fungsi ini akan terus diulang berdasarkan jumlah kedalaman pencarian (ply) yang telah ditentukan sebelumnya.

5. Perhitungan dan pemilihan nilai akhir.

Sebelum nilai terakhir dikembalikan, akan dilakukan dulu pengecekan apakah gerakan ini akan mengakibatkan terjadinya remis atau tidak.

Pengecekan tersebut dilakukan karena jika gerakan ini akan mengakibatkan lawan tidak mempunyai gerakan legal sama sekali tetapi kita tidak melakukan skak terhadap King lawan maka nilai akhir akan dirubah menjadi sangat besar, sehingga dalam proses selanjutnya gerakan ini tidak akan pernah dipilih, kecuali jika ini satu-satunya gerakan yang tersedia.

6. Eksekusi gerakan.

Setelah didapatkan index dari array dari proses sebelumnya maka akan dilakukan proses terkahir, yaitu perpindahan sesungguhnya dari bidak tersebut.

Setelah bidak benar-benar berpindah, maka giliran pemain tersebut sudah selesai dan akan dieksekusi kode terakhir sebagai berikut:

Setelah proses diatas dijalankan maka giliran akan berpindah ke pemain selanjutnya. Perulangan ini akan terus berlanjut hingga tiba saatnya permainan berakhir (Skakmat atau Remis).

IV. PEMBAHASAN DAN HASIL

Pengujian Move Generation

Pengujian ini bertujuan untuk mengecek validasi dari seluruh gerakan apakah sudah sesuai dengan peraturan catur atau belum. Pengujian ini dilakukan dengan metode piece-to-piece, seperti telah dijelaskan pada bab sebelumnya.

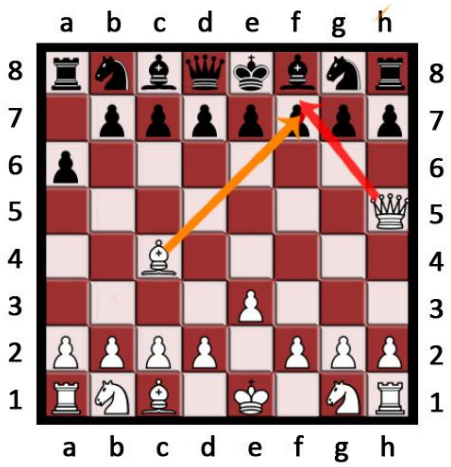
Pengujian Tingkat Kedalaman Pencarian

Pengujian ini bertujuan untuk melihat seberapa lama proses komputasi untuk setiap kedalaman pencarian. Lalu akan dilihat keefektifannya dalam bermain, karena jika hanya mengandalkan satu algoritma, semakin dalam pencarian tidak berarti kemampuan bermain akan semakin bagus.

Pengujian Kedalaman Pencarian Terhadap Pengambilan Langkah

Pengujian ini bertujuan untuk melihat apakah program sudah bisa bermain selayaknya pemain catur pemula, dikarenakan hanya dengan mengandalkan pencarian maka kemampuan bermain cukup terbatas.

Pengujian ini dilakukan dengan membuat beberapa skenario suatu pertandingan dengan jawaban terbaik yang sudah ditentukan. Selanjutnya kita tinggal melihat apakah gerakan yang akan dilakukan komputer sesuai dengan jawaban terbaik tersebut atau tidak.



Gambar 4. *Queen's Gambit*

V. KESIMPULAN DAN SARAN

Kesimpulan

Berdasarkan hasil dari perancangan, implementasi dan pengujian, maka diperoleh kesimpulan sebagai berikut :

1. Program dirancang dengan menggunakan algoritma Shannon Type-A, yaitu murni pencarian dengan evaluasi *minimax* tanpa ada tambahan algoritma lain.
2. Semakin besar tingkat kedalaman pencarian (*ply*) maka semakin lama proses komputasi
3. Namun semakin dalam pencarian, tanpa ada algoritma tambahan, bekum tentu komputer bisa bermain lebih baik.
4. Pencarian untuk seluruh gerakan dari awal hingga akhir tidak mungkin dilakukan dengan teknologi saat ini dikarenakan jumlah yang teramat banyak yang disebabkan oleh ledakan eksponensial.

Saran

Dalam perancangan dan pembuatan aplikasi ini masih terdapat kekurangan dan kelemahan, oleh karena itu masih diperlukan penyempurnaan untuk pengembangan kedepannya. Berikut adalah beberapa hal yang perlu diperhatikan dan disempurnakan:

1. Penambahan dan Penyempurnaan Algoritma agar bisa bermain lebih baik lagi.
2. Penggunaan hardware yang lebih baik untuk mempercepat proses komputasi.
3. Perbaikan *interface* agar program terlihat lebih menarik.

DAFTAR PUSTAKA

[1] [Shannon, 1950] Shannon, Claude. *Programming a Computer for Playing Chess*

[2] [PDF] Introduction - Historical look on chess programming - Top-5000.nl

[3] Chess Programming Wiki,"*http://chessprogramming.wikispaces.com/Home*."
[4] Chess, Wikipedia, "*http://en.wikipedia.org/wiki/Chess"*.

[5] Chess Flowchart, "http://www.chess.com/forum/view/general/chess-flowchart"